BIOSTATS 640 – Intermediate Biostatistics
Spring 2022

Introduction to R
01 – R Essentials

Welcome
In this introduction, you will learn how to download R and R-Studio, launch R-Studio, navigate among and use the panes in R Studio, issue some commands, and fix your mistakes!

__1. Download and Install R and R-Studio
__2. Launch R-Studio and Acquaint Yourself with its Interface
__3. Use Console as a Giant Calculator
__4. What Could Go Wrong
__5. Create Your First R Data
__6. A First Look at Your Data

#1. Download and Install R and R-Studio

Overview.

Step 1: Download and install R.
Step 2: Download and install R-Studio.
Step 3 (Mac Users Only): Download and install XQuartz

Mac Users

Preliminary (as you like): Consider watching a video for downloading and installing R and R-Studio (Duration: 3:01) https://www.youtube.com/watch?v=EmZqlcKkJMM

Preliminary (as you like): Download detailed instructions from course website (pdf, 4 pp) https://people.umass.edu/biep540w/pdf/HOW%20TO%20install%20R%20and%20R%20Studio%20MAC%20Users%20Fall%202021.pdf

Step 1: Download and install R.

1.1. Before you begin, check the version of the operating system on your mac by clicking on System Preferences
1.2 Launch the R Project for Statistical Computing, here: https://www.r-project.org/
1.3 At top, click on DOWNLOAD R, here: https://cran.r-project.org/mirrors.html
1.4 Choose your Cran Mirror. Suggestion: Scroll down and choose any of the ones from the USA (I picked Iowa State)
1.5 Click on DOWNLOAD R FOR MAC OS, here: https://mirror.las.iastate.edu/CRAN/bin/macosx/
1.6 Choose the ".pkg" file that matches your operating system.

Step 2: Download and install R Studio.
2.1 Launch the R Studio for IDE (IDE = *integrated development environment*) here:
https://www.rstudio.com/products/rstudio/download/
2.2 At left, under the column for R Studio Desktop FREE, click on the DOWNLOAD button here:
https://www.rstudio.com/products/rstudio/download/#download

2.3. After the download, open the ".dmg" file.
2.3 Movel the R Studio icon to Applications icon
2.4 Place a shortcut to R Studio on your dock.

Step 3: Download and install XQuartz.
3.1 Launch Xquartz, here: https://www.xquartz.org
3.2 Download, here: https://github.com/XQuartz/XQuartz/releases/download/XQuartz-2.8.1/XQuartz-2.8.1.dmg
3.3 Click on XQuartz.pkg and follow the installation instructions

WINDOWS Users

Preliminary (as you like): Consider watching a video for downloading and installing R and R-Studio (Duration: 3:01)
This one is for Windows 10 Users, just so you know:
https://www.youtube.com/watch?v=VLWaED9jTiA

Preliminary (as you like): Download detailed instructions from course website (pdf, 4 pp)
https://people.umass.edu/biep540w/pdf/HOW%20TO%20install%20R%20and%20R%20Studio%20WINDOWS%20Users%20Fall%202021.pdf

Step 1: Download and install R.
1.1 Launch the R Project for Statistical Computing, here: https://www.r-project.org/
1.2 At top, click on DOWNLOAD R, here: https://cran.r-project.org/mirrors.html
1.3 Choose your Cran Mirror. Suggestion: Scroll down and choose any of the ones from the USA (I picked Iowa State)
1.4 Click on DOWNLOAD R FOR WINDOWS, here: https://mirror.las.iastate.edu/CRAN/bin/windows/
1.5 In the first row, labelled BASE, at right click on INSTALL R FOR THE FIRST TIME, here:
https://mirror.las.iastate.edu/CRAN/bin/windows/base/
1.6 From the screen that appears, at the top click on the large font text DOWNLOAD R 4.1.2 for WINDOWS, here:
https://mirror.las.iastate.edu/CRAN/bin/windows/base/R-4.1.2-win.exe
1.7 After downloading, run the .exe installer

Step 2: Download and install R Studio.

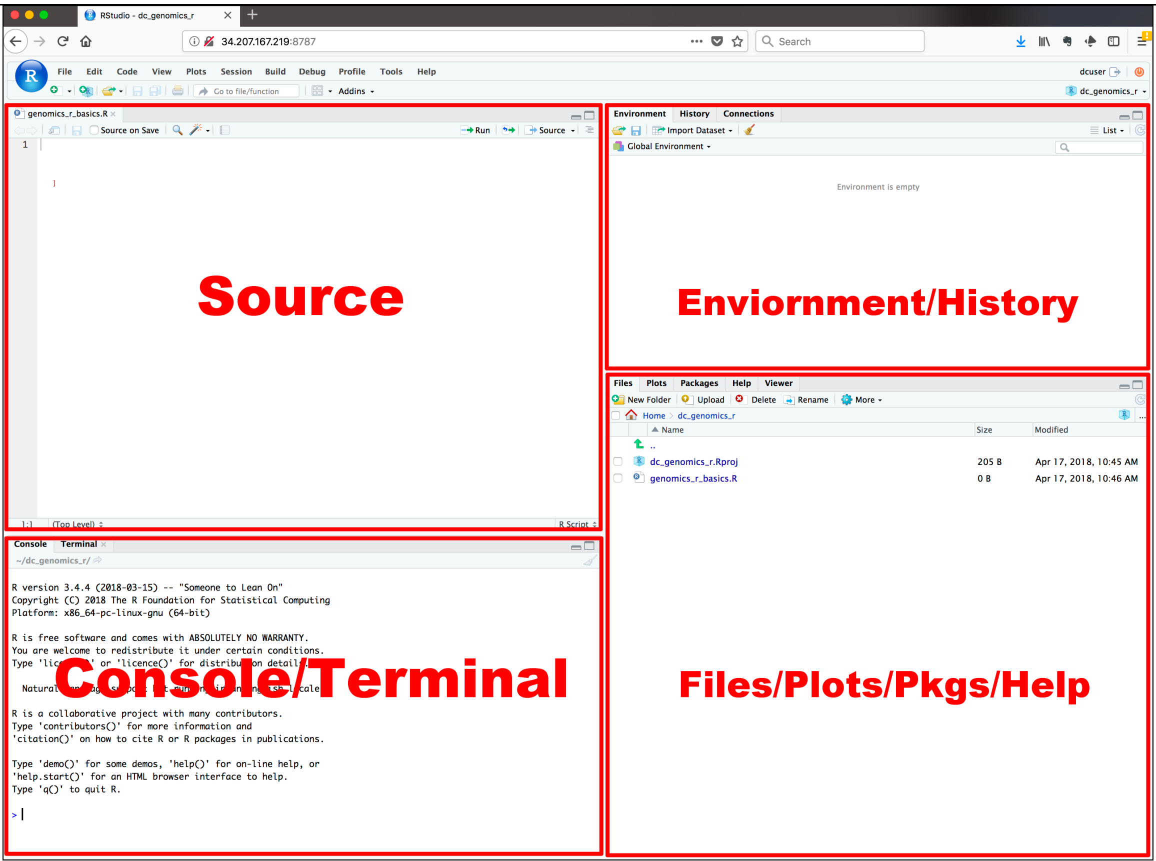
2.1 Launch the R Studio for IDE (IDE = *integrated development environment*) here:
https://www.rstudio.com/products/rstudio/download/
2.2 At left, scroll down to download R Studio for Desktop FREE, for Windows 10, here:
https://download1.rstudio.org/desktop/windows/RStudio-2021.09.2-382.exe

#2. Launch R-Studio and Acquaint Yourself with its Interface

Launch R Studio, not R.
We will be doing all our work in R Studio, ***NOT R***

Launch R Studio	NOT R

Acquaint Yourself with the R Studio Interface.
Consider visiting this introduction, here:
https://ismayc.github.io/rbasics-book/3-rstudiobasics.html



```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

(Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fdatacarpentry.org%2Fgenomics-r-intro%2F01-introduction%2Findex.html&psig=AOvVaw1hPui5N1bGL4CtSNUvA5Qi&ust=1600449355622000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCPiHvdLY8OsCFQAAAAAdAAAAABAs)

Quick Overview of the Panes in R Studio

Console/Terminal	• Default location: **lower left** • Code is executed from here • The prompt is a **">"** • IMPORTANT: Code typed into console is **NOT SAVED** • HACK: To retrieve previous command: **UP-arrow** • HACK: To clear window: **<control-l>** *this is the letter "el"* Good to know: No worries, your history is not lost
Source	• Default location: **upper left** • Here is where you will do your **R Script** and **R Markdown** work
Environment/History	• Default location: **upper right** • There are multiple tabs • **Environment tab**: Here you see all your stuff, called "objects" – datasets, variables, etc • **History tab:** Here you will see all your previous commands • HACK: To view your data: CLICK on its name
Files/Plots/Packages/Help	• Default location: **lower right** • There are multiple tabs. • Here you will find: plots, help w packages, importing from your computer, help

How to Move Between Panes

Shortcut	**Moves you to:**
< control > 1	Source/Editor (your script file)
< control > 2	Console
< control > 3	Help
< control > 4	History
< control > 5	Files
< control > 6	Plots
< control > 7	Packages
< control > 8	Environment
< control > 9	Viewer
< control > SHIFT 0	Returns you to original 4 panel display

#3. Use Console as a Giant Calculator

Welcome to the assignment operator, <-

Note: You could also use the equal sign, =, but this is not recommended

< - Translation: "Assign from the right to the left"

a < - 4 Example: - Create this thing (object) named 'a' and assign to it the number 4"

Comments, comments comments! #

Comments in R begin with a # R will ignore the rest of the line and continue its work at the start of the next line.

Tip – Make it a habit to comment your work. A lot! A year from now, when you look at your old work, you'll be so glad.

```
#  HOW TO create a vector object that is NOT SAVED
#  Use c() to create a variable (R calls this a vector object) - unsaved
c(1,2, 4, 8, 12, 13, 15)
## [1]  1  2  4  8 12 13 15

# HOW TO create a vector object that you DO SAVE and name as v1.
v1 <- c(1,2, 4, 8, 12, 13, 15)
# Bummer.  R Studio doesn't give you the result.
# You have to tell R Studio to show you something.

# HOW TO tell R to show you something
# To view the contents of an object, simply type the name of the object
v1
## [1]  1  2  4  8 12 13 15

# HOW TO obtain the data type (character, numeric, etc)
# Use class() to show the data type
class(v1)
## [1] "numeric"

# addition - Show the result but do not save it
4+6
## [1] 10

# Subtraction - Show the result but do not save it
4-6
## [1] -2

# Basic math in two steps:  (1) create the object y that is the solution (2) display the object y
y <- 4+6
y
## [1] 10

# Basic math in 2 steps connected by a semi-colon:  (1) create ;  (2) display
x<-5+8; x
## [1] 13
```

```
# Basic math in one step but now using parentheses to force R Studio to display
(x<-5+8)
## [1] 13

# Basic math with some annotation using paste( ) to produce reader friendly output
z<-8+16; paste("z = 8+16 = ",z)
## [1] "z = 8+16 = 24"
```

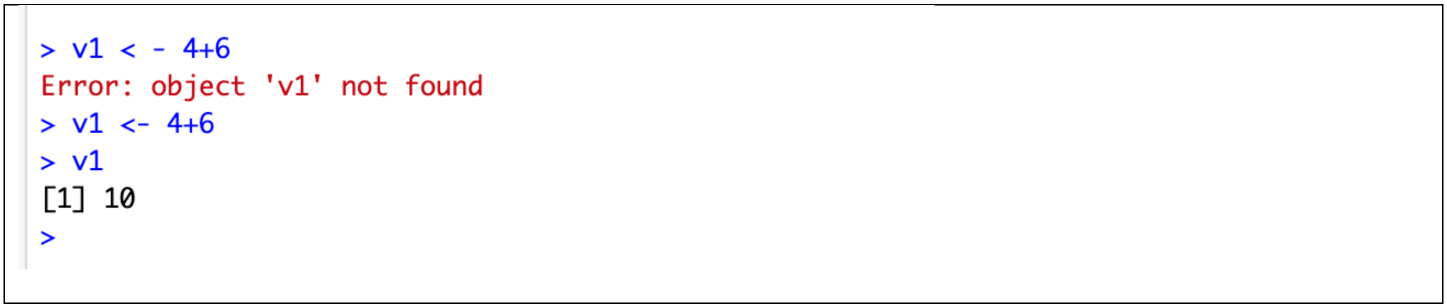
Mathematical Functions in R (partial listing)

Function	Definition	Example
+	Addition	> 2+2 [1] 4
-	Subtraction	> 5-3 [1] 2
*	Multiplication	> 5*4 [1] 20
/	Division	> 20/4 [1] 5
^	Exponentiation (raising to a power)	> 6^2 [1] 36
%/%	Integer part of division or quotient	> 48 %/% 5 What is whole number of 48/5? [1] 9
%%	Remainder part of division or quotient	> 48 %% 5 What is the remainder of 48/5? [1] 3
log()	logarithm to base e ("natural log") You may know this as ln( )	> log(34) [1] 3.526361 $e^{3.526361} = 34$
log10()	Logarithm to base 10	> log10(100) [1] 2 $10^2 = 100$
exp()	Exponentiation of the constant e Recall: e = 2.718 … (approx.)	> exp(4) [1] 54.59815 $e^4 = 54.59815$
sqrt()	Square root of	> sqrt(100) [1] 10 $\sqrt{100} = 10$
round(x,n)	Round x to the nth digit	

#4. What Could Go Wrong

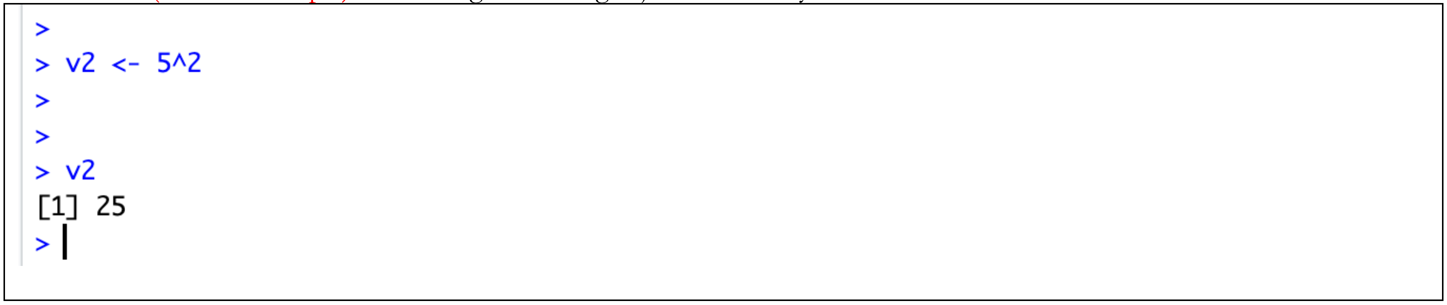
Invariably, in every R session you will ever do, you'll make mistakes or encounter glitches. I will try to anticipated these for you as best I can!

__1. **Error: My assignment operator did not work**
Solution (for this example): The assignment operator is <- with no space between the < and the -



```
> v1 < - 4+6
Error: object 'v1' not found
> v1 <- 4+6
> v1
[1] 10
>
```

_2. Error: I am not getting any result
Solution (for this example): Creating something is just that. Only. You have to tell R to then show it.



```
>
> v2 <- 5^2
>
>
> v2
[1] 25
>
```

_3. Error: I made a mistake several commands back; how do I fix that?
Solution (for this example): In the console window, do UP-ARROW repeatedly to access and then edit your command.

Hi – There's not really a screen capture I can do for you here.
Just try it!
- cb.

_4. Error: I am getting a + and no result

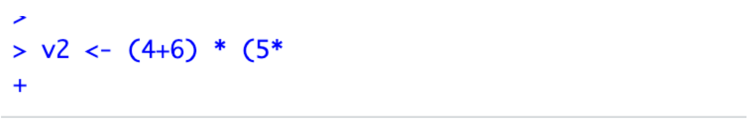
Solution (for this example): The + means that you have submitted a command that is incomplete. R is waiting for you to finish it. You have two options for a solution: (1) finish the command; or (2) abandon the command

SOLUTION #1 – Finish the command.
At the + simply finish the command and enter

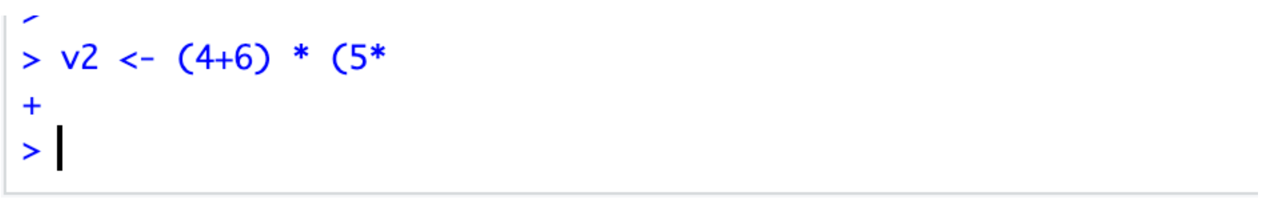
```
> (4+6)*(5
+
```

```
> (4+6)*(5
+ *7)
[1] 350
```

SOLUTION #2: Abandon the command.
Simply **Click <escape>** to return to the prompt



```
> v2 <- (4+6) * (5*
+
```



```
> v2 <- (4+6) * (5*
+
> |
```

#5. Create Your First R Data

Let's start with a really simple example:
Create an R dataset by combining columns (vectors)
Functions Used: c( ) and data.frame( )

In this example, we create a column of data for one numeric variable called **weight**. In R this is a vector of data type = numeric. We then create a 2nd column of data for one character variable called **town**. In R, this is a vector of data type = character. Finally, we combine these two variables (vectors in R) into a R dataset

```
> # create numeric vector called weight using c()
> weight <- c(161.3, 120.1, 223.2, 124.0, 88.2, 136.7, 140.0, 151.6)

> # create a character vector called town using c() with entries in quotes
> town <- c("amherst","amherst","hadley","amherst", "amherst","hadley","amherst", "amherst")

> # create an R dataset using data.frame( ) to combine vectors
> mydata <- data.frame(town,weight)

> # Show mydata
> mydata
```

```
     town weight
1 amherst  161.3
2 amherst  120.1
3  hadley  223.2
4 amherst  124.0
5 amherst   88.2
6  hadley  136.7
7 amherst  140.0
8 amherst  151.6
```

#6. A First Look at Your Data

In this introduction, we consider numerical summaries only. In a future introduction (after learning about the package ggplot2), we will learn ways to produce data visualizations. So, stay tuned! Pretty graphs are on their way.

In future introductions to R, we will be using packages to do all kinds of looks at data.
But here, we learn some simple ways to look at data using the functions that are included in your installation.

There's lots you can do, actually!

Welcome to the structure dataframe$variable
Yes, it seems a bit clunky but there you have it. In R, the convention for denoting a single variable in a dataframe is the notation: dataframe$variable.

dataframe$variable

no spaces
around the $

- **"dataframe"** is analogous to "dataset" in SAS or Stata or Minitab, etc
- **"dataframe"** is analogous to "sheet" in an Excel file that has data in a sheet

- **"variable"** is just what you think it is. It is analogous to "variable" in SAS or Stata or Minitab, etc
- **"variable"** is analogous to "column" in an Excel file that has data in a sheet

Example

mydata$town
Dataframe name: mydata
Variable name: town

Missing Values in R

NA

Note: There are other ways to denote missing values in R; we will get to these in a future introduction

Preliminary: Use str( ) to determine the data type for each variable in your dataframe
Why? R will do some descriptives for some datatypes but not others.
For example, R will not produce descriptives for a variable that is of data type = character.

```
> str(mydata)                                          # examine structure of dataframe
```

```
'data.frame':	8 obs. of  2 variables:
 $ town  : chr  "amherst" "amherst" "hadley" "amherst" ...
 $ weight: num  161.3 120.1 223.2 124 88.2 ...
```

Key:

- This is a dataframe
- There are 8 observations (sample size n=8)
- There are 2 variables: town and weight
- town is of data type = character
- weight is of data type = numeric

As needed: Use factor( ) to create a categorical variable from your character variable.
Important: R uses a different name for "categorical variable". R calls these factors
We will learn a lot more about factors in future R introductions!

```
> mydata$townf <- factor(mydata$town)                 # create a new variable called townf
> str(mydata)                                         # check that the new variable is there and correct
```

```
'data.frame':	8 obs. of  3 variables:
 $ town  : chr  "amherst" "amherst" "hadley" "amherst" ...
 $ weight: num  161.3 120.1 223.2 124 88.2 ...
 $ townf : Factor w/ 2 levels "amherst","hadley": 1 1 2 1 1 2 1 1
```

Summary statistics for a continuous variable using summary( )

```
> summary(mydata$weight)
```

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   88.2   123.0   138.3   143.1   154.0   223.2
```

IMPORTANT TO REMEMBER: R does *not* produce descriptive statistics for a character variable

Why: A character variable has responses that are simply strings; it's not possible to produce descriptives of strings!

```
> summary(mydata$town)
```

```
  Length     Class      Mode
       8 character character
```

Frequencies for a categorical variable (remember: R calls this a factor) using summary( )

```
> summary(mydata$townf)
```

```
amherst  hadley
      6       2
```

Numerical Summary for EVERY variable in your dataframe using summary( )

```
> summary(mydata)
```

```
     town               weight          townf    
 Length:8           Min.   : 88.2   amherst:6  
 Class :character   1st Qu.:123.0   hadley :2  
 Mode  :character   Median :138.3              
                    Mean   :143.1              
                    3rd Qu.:154.0              
                    Max.   :223.2              
```

Key:

- Notice how NO descriptive statistics are produced for the character variable town
- As a general rule, do NOT overwrite an existing variable. Always preserve your original data. Here, I created townf as a new variable, leaving the source variable town UNchanged

Illustration: How to Calculate a Statistical Summary (e.g., mean, variance, etc)

```
> mean(mydata$weight)
```

```
[1] 143.1375
```

What Could Go Wrong: How to Calculate a Statistical Summary (e.g., mean, variance, etc)

Error: My calculation of a statistic produced NA

Solution (for this example): You need to tell R to exclude missing values NA in the calculation

```
> age <- c(33,12, NA, 67, 82, 91)
> mean(age)                             # calculate mean
[1] NA
```

```
> mean(age,na.rm=TRUE)                  # calculate mean with option na.rm=TRUE to remove missing values
[1] 57
```

Some Statistical Functions in R

Function	Definition	Example

length(x)	Number of values in vector x	`x <- c(3,1,6,0,6)`
		`> length(x)`
		`[1] 5`
		
		`… alternatively, you could do ..`
		`> length(c(3,1,6,0,6))`
		`[1] 5`

max(x)	Maximum of values in vector x	`> x <- c(3,1,6,0,6)`
		`> max(x)`
		`[1] 6`

min(x)	Minimum of values in vector x	`> x <- c(3,1,6,0,6)`
		`> min(x)`
		`[1] 0`

mean(x)	Mean of values in vector x	`> x <- c(3,1,6,0,6)`
		`> mean(x)`
		`[1] 3.2`
		`> x <- c(3,1,NA,0,6)` *`Oops a missing!`*
		`> mean(x,na.rm=TRUE)`
		`[1] 2.5`

median(x)	Median of values in vector x	`> x <- c(3,1,6,0,6)`
		`> median(x)`
		`[1] 3`

quantile(x,c(.25,.75))	Obtain 25th and 75th quantile values in vector x	`> x <- c(3,1,6,0,6)`
		`> quantile(x,c(0.25,0.75))`
		`25% 75%`
		`  1   6`

range(x)	Display minimum and maximum values in vector x	`> x <- c(3,1,6,0,6)`
		`> range(x)`
		`[1] 0 6`

sd(x)	Standard deviation of values in vector x	`> x <- c(3,1,6,0,6)`
		`> sd(x)`
		`[1] 2.774887`

sum(x)	Total of values in vector x	`> x <- c(3,1,6,0,6)`
		`> sum(x)`
		`[1] 16`

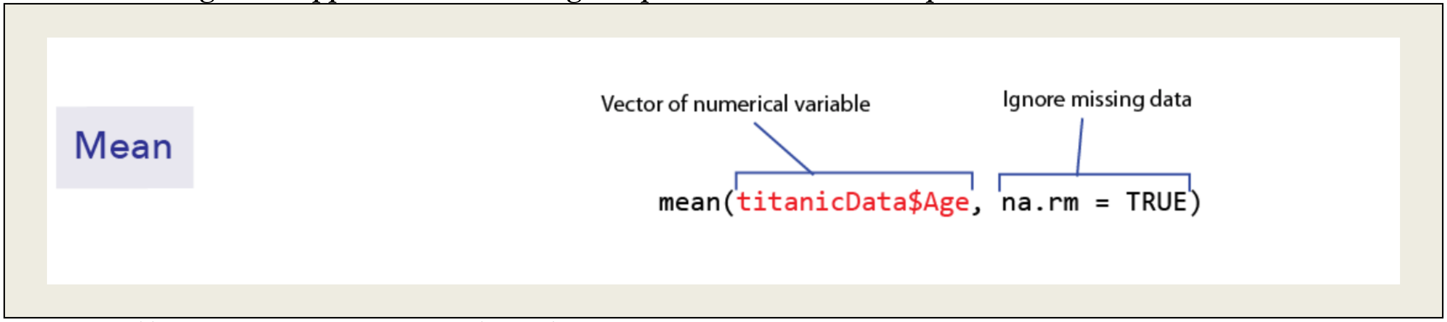
var(x)	Variance of values in vector x	`> x <- c(3,1,6,0,6)`
		`> var(x)`
		`[1] 7.7`

abs(x)	Absolute values of values in vector x	`> abs(2-10)`
		`[1] 8`

factorial(x)	Calculate
	x! = x(x-1)(x-2)…(2)(1)
		`> factorial(4)`
		`[1] 24` `4! = 4*3*2*1 = 24`

rank(x)	Ranks of values in vector x	`> x <- c(3,1,6,0,6)`
		`> rank(x)`
		`[1] 3.0 2.0 4.5 1.0 4.5`

Recommended general approach for obtaining sample statistics is to use option na.rm=TRUE or na.rm=T



```
mean(titanicData$Age, na.rm = TRUE)
```

Source: https://whitlockschluter3e.zoology.ubc.ca/RLabs/R_tutorial_Describing_data.html